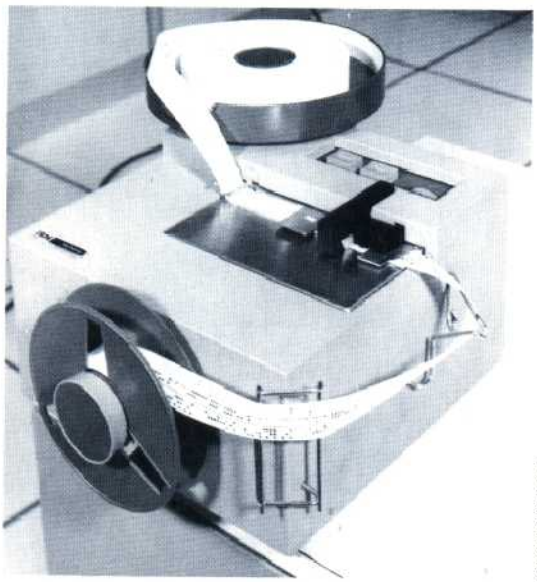
ASCII

The American Standard Code for the Interchange of Information is the system adopted by most microcomputer and software manufacturers for representing alphanumeric characters in binary form. There is no fundamental reason why the letter A should be represented by 01000001 (as it is in this system), since the CPU is not aware of whether it is processing letters, numbers or symbols. The main benefit of agreeing on a standard code is that programs can then be written to run on several different machines. Furthermore, data can then be sent from one computer to another, either by means of a telephone link or stored on a magnetic medium.

ASCII is the most popular system, but by no means the only one. Another is EBCDIC – the Extended Binary Coded Decimal Interchange Code – which was favoured by IBM on its early mainframe machines, though not the IBM PC.

Many standard commercial packages for microcomputers have the facility to produce *ASCII files*, the functions of which are best illustrated by example. A document stored as a file from a word processing package will usually contain a number of special control characters — to indicate that a heading should be centred on a page, for example. If an ASCII file is generated, all these control characters will be removed, and replaced with pure ASCII characters, such as the correct number of spaces. An ASCII file can then usually be read in to another word processing package that perhaps uses a different set of control characters.

ASCII Code
The best place to see a coding system like ASCII in action is on an old-fashioned paper tape machine, where each character is represented by a row of holes across the tape



COURTESY OF IBM

ASSEMBLY LANGUAGE

Assembly language, in essence, is a more readable way of expressing machine code, using alphanumeric labels instead of hexadecimal addresses and mnemonics instead of numbers for the opcodes. However, unlike other higher-level languages, Assembly language (sometimes confusingly referred to as 'assembler') translates byte-for-byte into machine code. Consequently, there is no loss in efficiency through writing in Assembly language first. Once the program is complete, it is passed to the assembler program, which translates the Assembly language labels and mnemonics into their machine code equivalents. The Assembly language program (known as the source code) is filed for reference, while the machine code translation (known as the object code) is loaded directly into RAM for immediate execution and/or filed for later use.

ASYNCHRONOUS

There are two types of data transmission – synchronous and asynchronous – and the terms are most commonly encountered when referring to serial data transmission over a long distance, usually by means of a telephone line. *Asynchronous transmission* is the simpler to implement, and is the basis for interfaces such as the RS232, which are found in many home computers. Synchronous transmission, however, is the more efficient in terms of maximum data transfer rate, and is used on all large computer systems.

In an asynchronous system, transmission commences whenever the sending device is ready to issue the data. Each byte is preceded by a *start bit* — just an additional bit at the start that 'wakes' up the receiving computer and tells it to expect a further eight bits of real data, at the baud rate that has been set on both devices. The data is usually followed by one or two *stop bits*, which allow the receiver to 'collect its thoughts'. This pattern is followed for every byte (i.e. every character) transmitted.

On a synchronous system, both devices feature very accurate clocks, which are synchronised, and transmissions occur only at specified intervals. It's a bit like saying to someone: 'I will be waiting by the telephone at the start of every hour for five minutes in case you call' — although in the computer the interval will be in terms of microseconds. The result is that the start and stop bits are no longer needed and efficiency is thus increased. The main difficulty with synchronous transmission is the need to keep both clocks in perfect synchronisation.

ATTRIBUTE

The screens on early microcomputers could only display text in upper and lower case (along with a few graphics symbols if you were lucky). Now some quite elementary home computers allow the programmer to specify each character's colour, the background colour on which it is printed, whether it is normal or extra bright in intensity, and whether steady or flashing. These specifications are called the *attributes* of a character, and they are usually stored in a single byte of RAM for each screen location. Each bit, or group of bits within that byte, will look after one of those attributes, and they can either be determined using the BASIC commands such as BRIGHT, FLASH, INK, PAPER and so on, or by manipulating the bits directly using the AND and OR functions of machine code.